10/523270

\\RRT

DT05 Rec'd PCT/PTO 31 JAN 2005

# A Scheduling Method for Polling Device Data

## Field of the Technology

The invention relates to network management field, in particular to a dispatching method for polling device node data in a network management system.

## Background of the Invention

In a network management system based on Simple Network Management Protocol (SNMP), it is necessary to get message for state changes and configuration changes of device nodes in the network in time. Many network management systems apply a method for periodically reading device data, i.e., for polling device data. When there are a huge number of devices, the polling operation will expense a lot of network resources, so a dispatching method for the polling is necessary.

At present, the dispatching methods usually take a device as a unit or a type of devices as a unit for dispatching. When a device is taken as a unit, a polling period is set for each device, data of the device are read once every period and introduced into the network management system. When a type of devices are taken as a unit, a polling period is set for each type of devices and every period devices of the type are polled sequentially for reading data of each device.

When there are small number of devices, such as several devices or tens of devices, the prior dispatching method works well and can be simply implemented. Nevertheless, when there are huge number of devices, such as hundreds and thousands of devices, many problems will appear.

When a device is taken as a unit for dispatching, first, control for every device will occupy a lot of system resources, so the system is over loaded and normal operation of other network management functions will be affected. Secondly, it is impossible to make coordination between devices, so it is possible that a lot of devices are polled in the same duration, which will cause a burst network flow and system load on the network and in a serious situation the network will be blocked. In this way, the network management system will collapse due to overload, normal running of network service will be affected in the case that inner-band network management is used. Thirdly, if there are a lot of devices, independent configuration for polling period of each device will greatly increase work load of network administrator, and increase the difficulty of management and operation.

When a type of devices are taken as a unit for dispatching, the burst flow caused by polling management information is controlled in a certain degree and the configuration of polling period is simple, but since all devices are sequentially polled, the polling period is long and the real time response ability is worse, system resources and network bandwidths cannot be utilized sufficiently. When users demand the change of network management system should have a high real-time ability, this dispatching method cannot satisfy the requirement. In addition, there is little coordination between various types of devices, and all devices are processed jointly, so individuation cannot be realized.

Besides, changeability of data on a device is different but the prior dispatching method reads all data on a device with the same period, so various data on the same device cannot be control individually. Therefore, it is impossible to have a shorter period for the data that change frequently and a longer period for the data that change rarely.

In addition, the prior dispatching methods can only control the polling period but cannot control the polling time, so it is impossible to distribute the polling to the idle time of services on the network in the case of inner-band network management.

## Summary of the Invention

In order to overcome the problems of the prior art, an object of the invention is to provide a dispatching method for polling device data.

A dispatching method for polling device data according to the invention comprises the steps of:

1) sorting managed devices according to their types, sorting various types of data of each device so as to form different modules, and assigning a priority attribute and a polling period attribute to each module;

2) dividing the managed devices into two sets: one set consisting of devices to be polled and the other set consisting of devices whose connection states need to be detected; and

3) polling each module in the set consisting of devices to be polled according to its priority and polling period periodically.

In the invention, step 3) may further comprise:

forming a current polling task queue according to said periodical polling, and dispatching the polling through the current polling task queue; wherein the data items for describing the current polling task queue include task ID, occupied flag, module ID, device ID, activation time and priority; said activation time is the current time in the case of inserting a task and will be updated when a report about executing situation of the task sent from daemons has been received; said occupied flag is set free after a corresponding message showing the polling task has been completed is received or the polling task is overtimed.

The invention may further comprise the step of setting a maximum number of polling tasks. Here, the current polling task queue is generated according to the maximum number of polling tasks.

The invention may further comprise the step of setting a polling initiating time for system. Here, the periodical polling is implemented based on the polling initiating time plus a polling interval.

The polling period attribute of a module mentioned above can be a multiple of a polling interval, equaling to a multiple of the interval between the periodical system polling in step 3).

The above-mentioned set of devices to be polled can be a current operation device set and the set of devices whose connection states need to be detected can be a current display device set. Here, the step of dispatching the polling through the current polling task queue comprises:

a. setting the polling initiating time at the summation of the current time plus a polling interval;

b. determining whether there is a free task in the current polling task queue based on the occupied flag; if so, continuing the process, otherwise returning to step b;

c. selecting the next device module to be polled from the current operation device set; and

d. determining whether the information obtained in step c is Null or not; if not, assigning a task ID to the selected device module and inserting the task ID into the current polling task queue, and simultaneously sending a message for initiating the polling of said device module to the corresponding daemon process, then returning to step b; if so, determining whether all tasks in the current polling task queue are in free state, if all tasks are in free state, ending the process, otherwise returning to step b.

The method may further comprise:

4) selecting sequentially a device from the set consisting of devices whose connection states need to be detected and making ping operation for the device; wherein the success of ping operation shows the device is connected to the network management system and failure of ping operation shows the device is not connected to the network management system; if the connection state of said device is changed, notifying other daemons and foregrounds about this condition.

In this invention, all devices are divided into groups according to their types and inner modules, and assigned a corresponding priority and polling period, so various type of data of the same device can be controlled individually based on their changeability. For example, the data changed more often can be polled more frequently, and the data changed less often can be polled less frequently.

In this invention, the managed devices are divided into two sets: a set of current display devices and a set of current operation devices. In this way, only the devices being operated by users are polled. Besides, different data of the same device is individually set with different polling periods based on the data changeability. The number of polling tasks can be adjusted based on the configuration of network management system and network bandwidth so that the system resources can be utilized reasonably and all polling can be reasonably dispatched.

In this invention, all devices in network management system are dispatched jointly. In addition, device data and connection state of device are processed individually. Accordingly, the total system load and network load of the network management system can be controlled effectively.

## Brief Description of the Drawings

Figure 1 is a diagram illustrating the architecture of a network management system.

Figure 2 is a flowchart illustrating selection of a polling module.

## Detailed Description of the Invention

The invention will be described in more detail hereinafter with reference to the drawings.

With the invention, all managed devices in the current network management system are sorted according to their types, various types of data in each device are sorted so that various modules are formed; and then every module is assigned a priority attribute and a polling period attribute. Among them, the priority shows that which polling sequence should be taken for device data. If two modules have the same priority, then it shows that there is no sequence relationship between them, i.e. either one is permitted to be polled first. The polling period represents a polling interval of the module. Consequently, the invention can set different polling periods for different kinds of data of the same device and take different polling policies according to the changeability of data. For example, the data changed more often can be polled more frequently, and the data changed less often can be polled less frequently.

Furthermore, with the invention, all the managed devices are divided into two sets: one is the set consisting of devices that need to be polled, and the other is the set consisting of devices whose connection states need to be detected. For the former, all data is polled periodically, and for the later Ping operation is executed periodically but without reading the device data. Therefore, it is possible to change the polling policy in real time and flexibly based on the device situation, for example, taking priority for polling the current running devices. If a device is not at the current active window, but it is desired that the device should be polled too, then it is necessary only to add the device in the set consisting of devices that need to be polled at the daemon.

By dividing all the managed devices into two sets in the invention, all devices in the network management system can be dispatched jointly, and joint dispatch for polling tasks can be provided according to the possible maximum number of polling devices and the bandwidth of the network, so the total system load and network load of the network management system can be controlled effectively, and the system resources can be used more reasonably.

In the following the invention will be described with reference to an embodiment. It should be understood that the embodiment is only for description of the invention and is by no means to limit the protection scope of the invention, but, on the contrary, the invention is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

In general, a network management system consists of many subscriber terminals and many server programs (daemon processes); the invention adds a specialized dispatching program of polling, i.e. daemon process 4, in the original architecture, as shown in Fig.1. Obviously, other system structures can also be used in the invention.

The following data structures are defined for the dispatching process of polling:

1. Device type description

| Device type | Module ID | Priority | Polling interval multiple | Corresponding daemon ID |
|---|---|---|---|---|
| | | | (default value is one) | |
| | | | | |

In the table, the combination of the device type and module ID is a unit index. The polling interval multiple is a polling period attribute of the module that is the ratio of the module polling period and the system polling period (basic polling period). For example, if the polling interval multiple of a module is two, then the polling period of the module is double of the polling period of a module whose polling interval multiple is one.

2. Current operation device set (OperationList)

| Device ID | Module ID | Device type | The last polling time |
|---|---|---|---|
| Device 1 | Module 1 | | Sorting by this value in memory |
| Device 1 | Module 2 | | |
| Device 1 | Module 3 | | |
| Device 1 | Module 4 | | |
| Device 2 | | | |
| Device 3 | | | |
| ... | | | |

In the table, the combination of the device ID and module ID is a unit index.

3. Current display device set (DisplayList)

| Device ID | Connection state |
|---|---|
| | |
| | |
| | |

In this embodiment, the set consisting of devices that need to be polled and the set consisting of devices whose connection states need to be detected are the OperationList and DisplayList respectively. The DisplayList is the set of devices that can be found by subscribers; if other windows in an interface hide information of a device, then the device does not belong to this set. The OperationList is a set of devices that are included in an interface that is being operated by subscribers, i.e. the devices included in an activated window at the subscriber end. The above-mentioned two sets are reported to the server by subscribers, and the server will summarize all of them to form DisplayList and OperationList of the whole network management system.

4. Current polling tasks queue

| Task ID | Occupied flag | Module ID | Device ID | Activation time | Priority |
|---|---|---|---|---|---|
| 1 | [busy/free] | | | | |
| 2 | | | | | |
| 3 | | | | | |
| ... | | | | | |
| n (n = maximum number of polling tasks) | | | | | |

Each time when initiated, the polling task is put in the above table with a busy flag and an initiating time. The busy flag will be released after the task completion information is received or the task is overtimed. Whether a task is overtimed is determined through the interval between the current time and the activation time. If the interval excesses the polling overtime that is pre-set in the global data below, the task is overtimed. The activation time is set as the current time when the task is initiated, and then, when a report of executing situation for the task sent by the other daemon processes periodically is received, the activation time will be updated.

5. Global data

| | |
|---|---|
| Polling interval | It shows the expected interval between this initiating time and the next initiating time for polling of the whole system; if polling all devices once needs more time then this one, then devices cannot be polled with this interval. |
| Initiating time of polling (hours/minutes/seconds) | In memory, it is initiated to the absolute time with information of current year, month and day. |
| Maximum number of polling tasks | It shows the maximum number of simultaneous polling tasks supported by the network management system. |
| Polling overtime period | When other daemon processes have received a polling command from the dispatching process, they need report periodically to the dispatching process whether the polling task is completed; if there is no report at overtime, the dispatching process will consider that the operation is ended. |
| Ping overtime period | |
| Ping repeated times | |

Based on the data mentioned above, the polling tasks are dispatched with the following steps.

In step 1, firstly, all kinds of parameters are initiated; description data of device types and polling overtime period are read. Then, DisplayList and OperationList are generated through interfaces of other daemon processes; current polling task queue is generated based on the maximum number of polling tasks. If there is an initiating time of polling, then the timer is set as to initiate the dispatching thread for polling at the initiating time, otherwise the timer is set as to initiate the dispatching thread immediately and the process goes to step 2. Meanwhile connection state detecting thread is initiated, and the process goes to step 3.

In step 2, every module of every device is polled in turn according to its polling priority and polling interval multiple. Specifically, the step comprises the following steps:

a. setting initiating time of polling at the summation of the current time plus the polling interval;

b. determining whether there is a free task in the current polling task queue; if so, then continuing the process, otherwise returning to step b;

c. selecting the next unit (device ID + module ID) to be polled from the OperationList, wherein the step comprises the following specific steps as shown in Fig.2:

c1. selecting the next unit from the OperationList;

c2. determining whether [(the current time – the last polling time)/polling interval multiple of the module] is greater than or equal to the polling interval; if so, executing step c3 continuously, otherwise going to step c4;

c3. determining whether there is a unit with higher priority being polled in the different module polling tasks of the current same device; if so, returning to step c1, otherwise returning the unit information and ending step c;

c4. determining whether the polling interval multiple of the unit is greater than one; if so, returning to step c1, otherwise returning a NULL to show that no any module needs to be polled and ending the step c.

d. when the information returned in step c is a NULL, determining whether all items in the OperationList are in free state; if so, it showing that one polling for the system is ended and ending step 2, otherwise returning to step b.

e. when the information returned in step c is not a NULL, assigning a task ID to the selected unit, then inserting the task ID in the OperationList; at the same time sending information for initiating a polling module of the device to the corresponding daemon process and returning to step b.

In step 3, devices are selected in turn according to the DisplayList and Ping operations are made for the selected devices. If connection state of the device has been changed, then other daemons and foregrounds will be informed.

During the process of selecting the next unit to be polled from the OperationList in step c, the dispatching process also receives reports about whether a polling task is ended from other daemons. If the polling task is ended, the occupied flag of the task is released in the list of current polling tasks and the task is set free. If the polling task is not ended, the activation time of the task is updated to the current time. If the task is overtimed, the occupied flag of the task is released and the task is set free also.